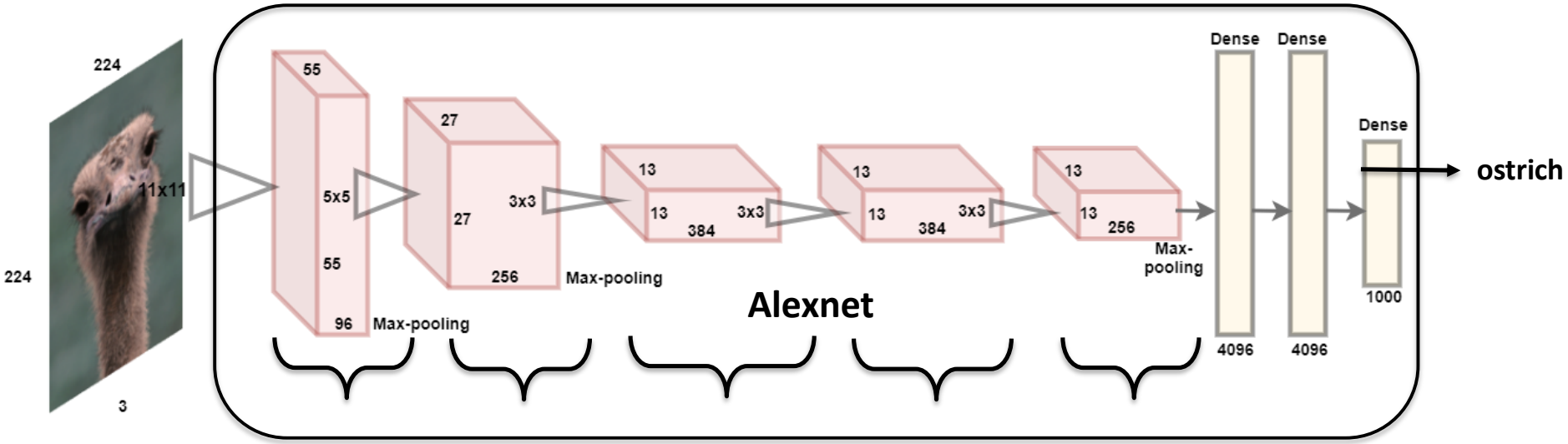
Visualization and Understanding of Deep Neural Networks

Vinkle Srivastav

UNIVERSITÉ DE STRASBOURG

ICUBE

Objectives



We will try to understand
how predictions are made by a trained model

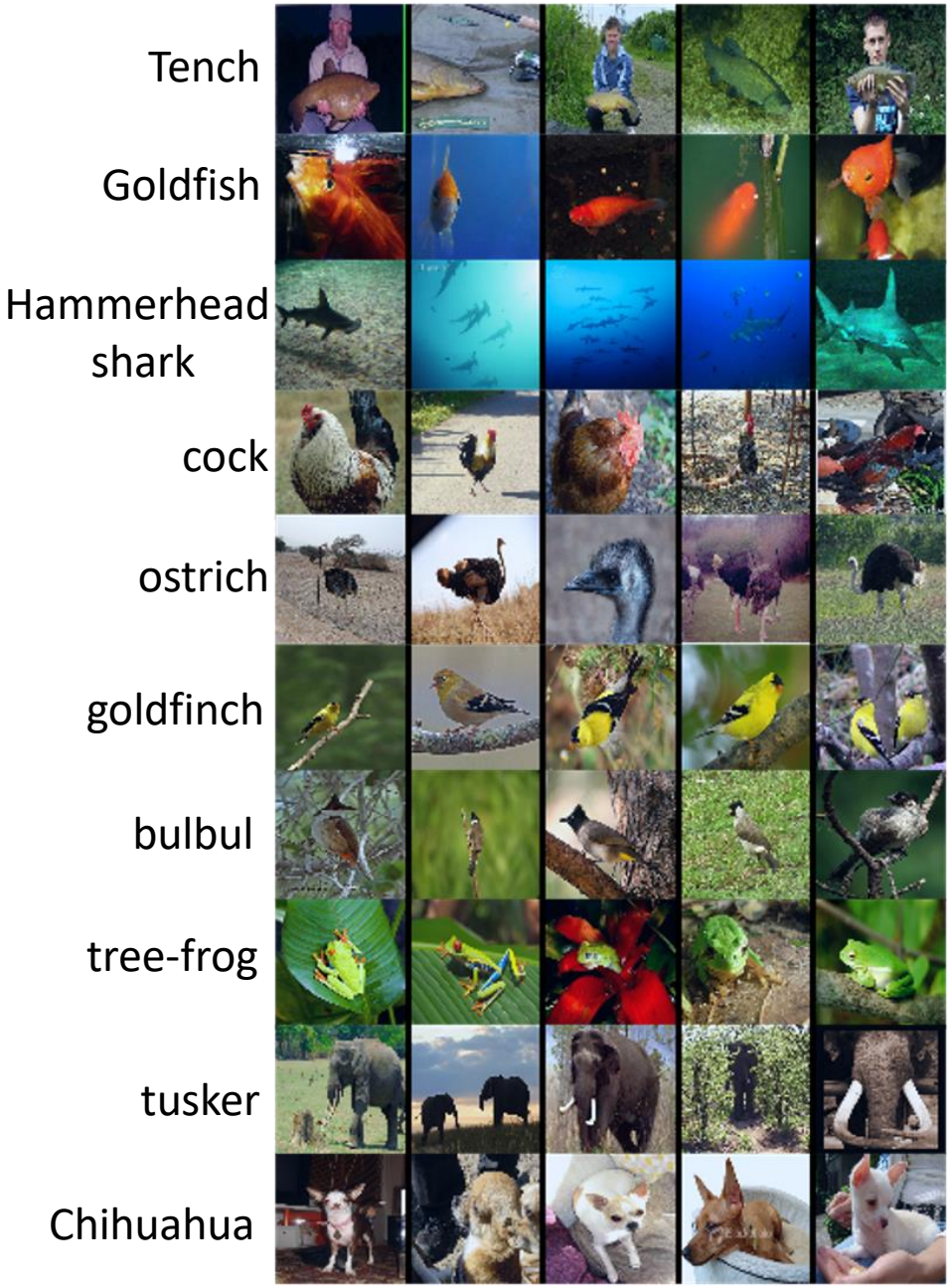
Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.

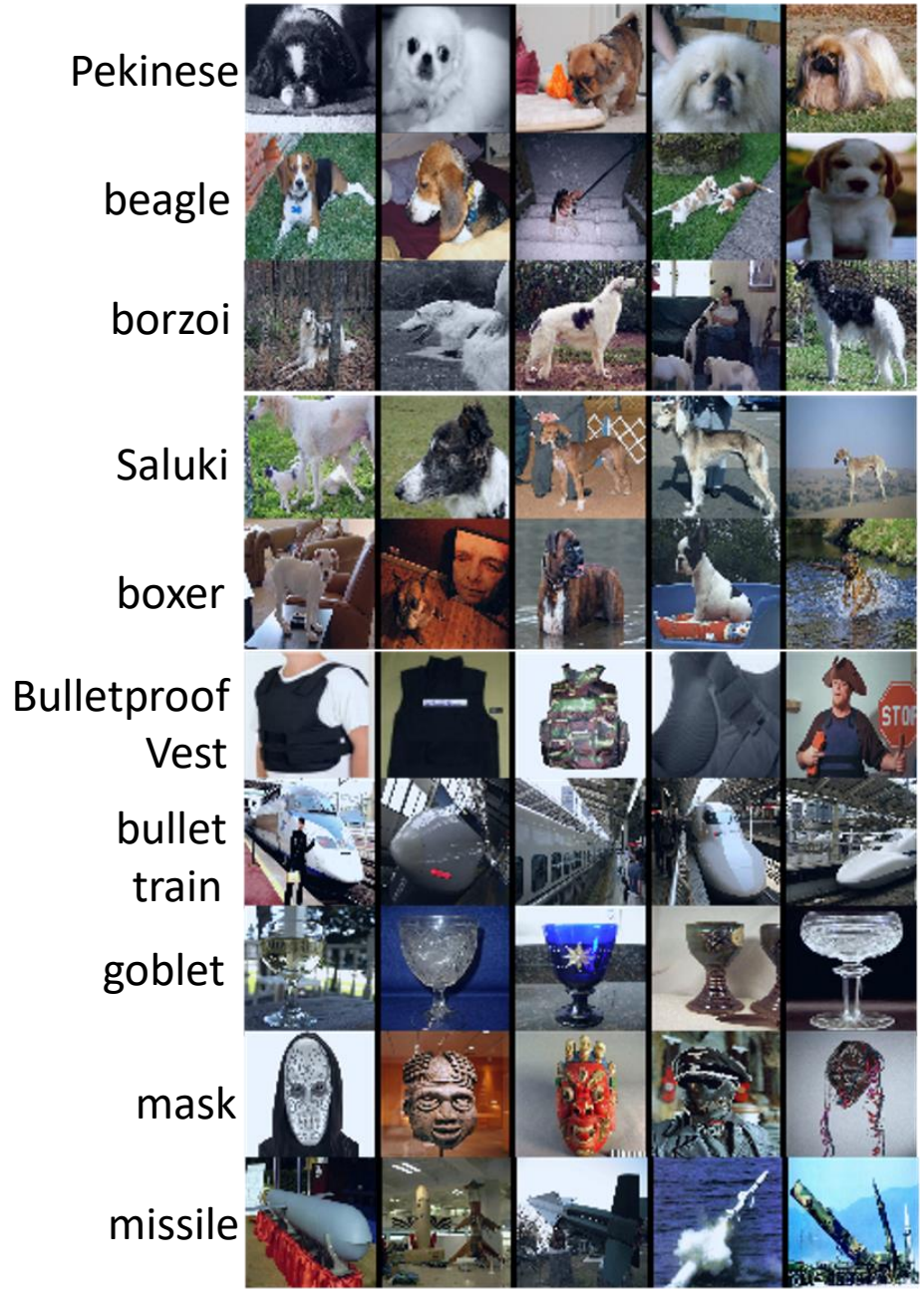
Objectives

- Visualize the first layers
- Interpret the last layers
  - K Nearest Neighbors (KNN) to visualize the images in feature space
  - Principal Component Analysis (PCA) to project features on 2D space
- Visualize image pixels that affect the classifier output
  - Saliency maps
    - via occlusion / via gradient / via smooth gradient
  - Class Activation Maps
  - Gradient Class Activation Maps
- Visualize intermediate features
- Visualize per-class features



Meet Our Experimental Dataset (Sample Images from Imagenet)

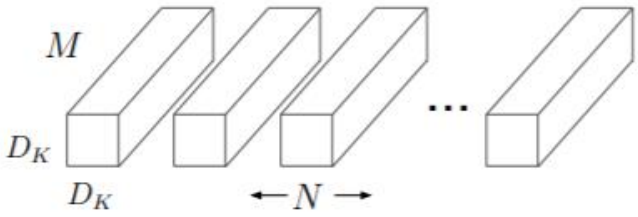




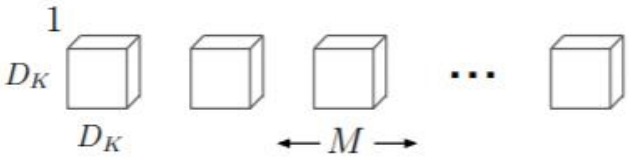
Experimental dataset for this part
496 images from 20 classes from Imagenet dataset (approx. 25 images/class)

Meet Our Experimental Model (Mobilenet-V2)

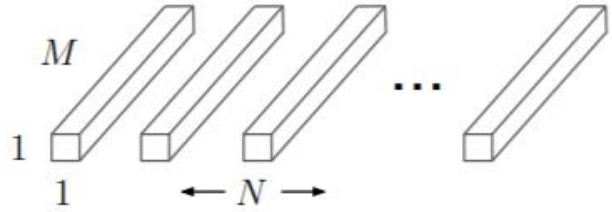
MobileNet-V2



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters

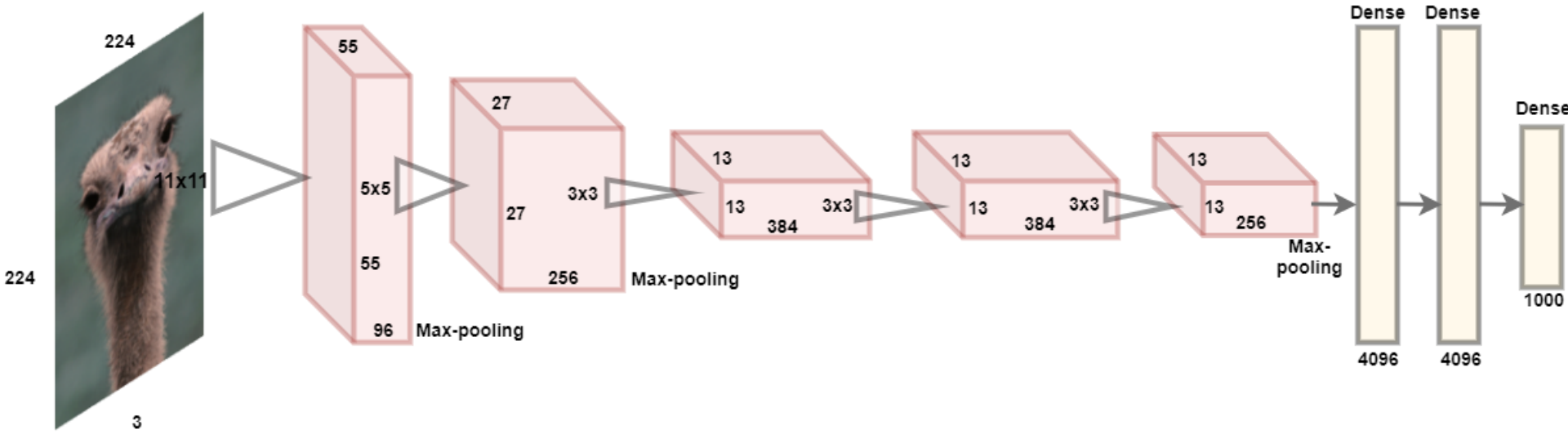


(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

- Depthwise convolutions
- Conv kernels (D*D*n_channels) replaced by 2D conv (D*D*parameters) followed by 1D conv in the channel dimension (n_channel parameters)

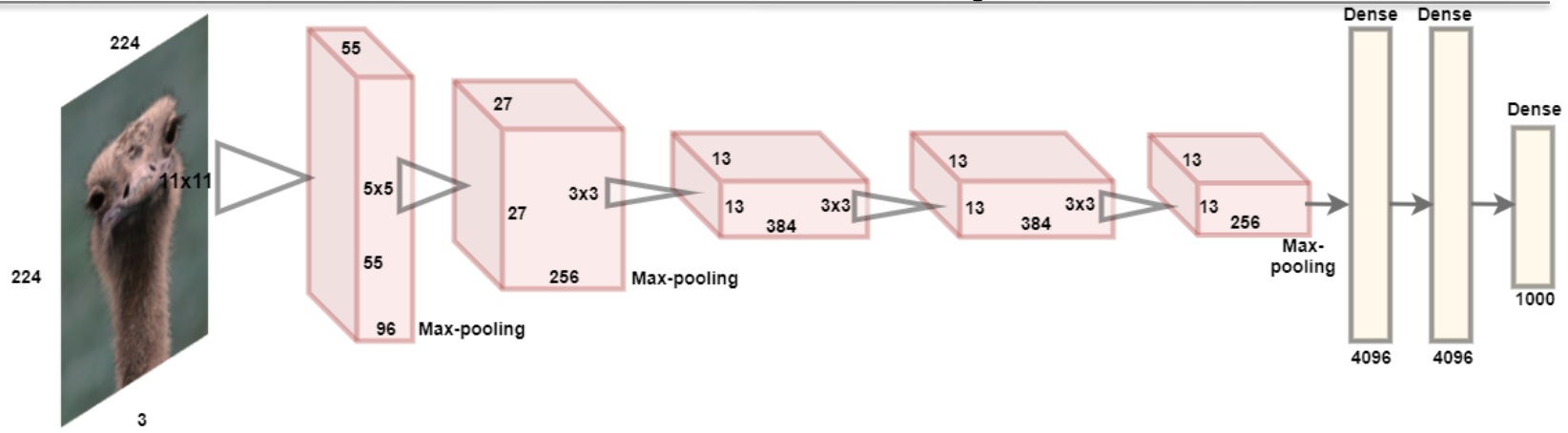
Alexnet

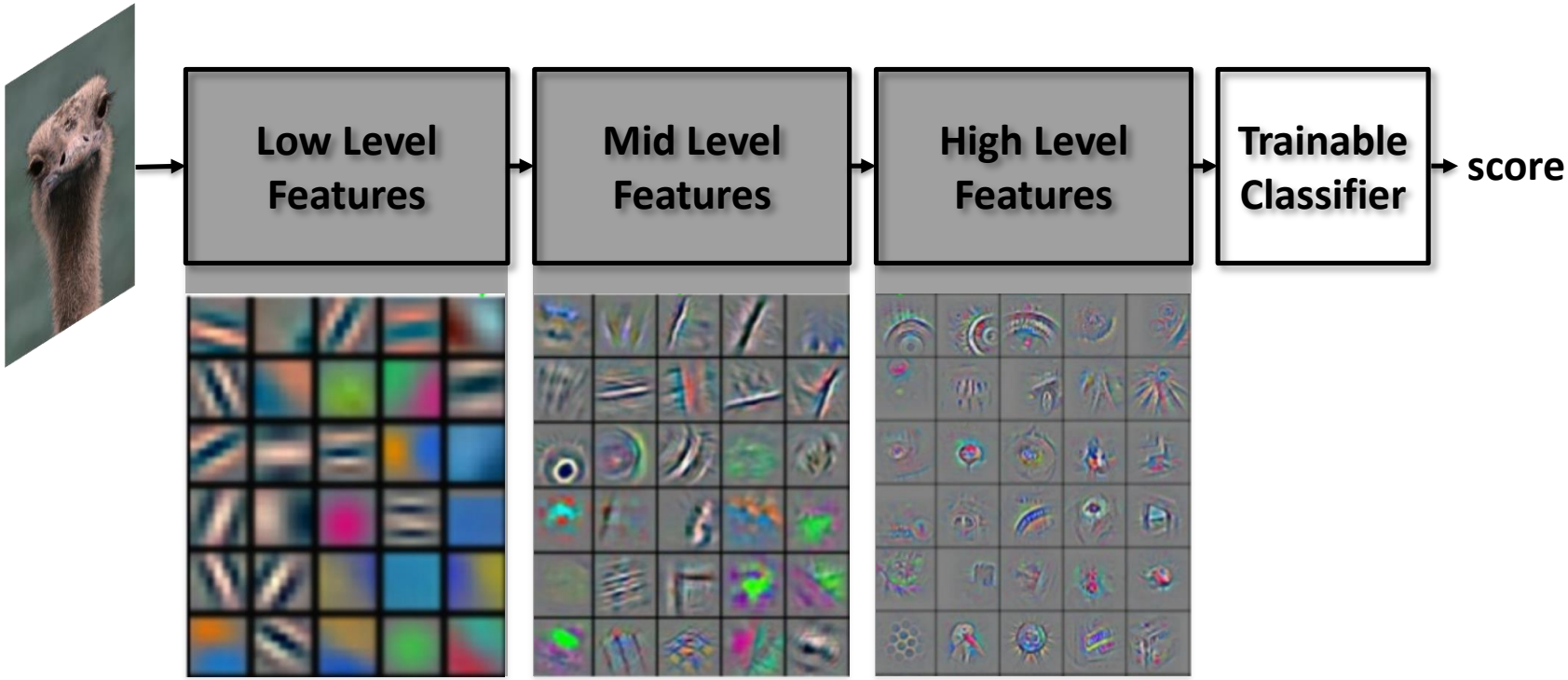


For simplicity, we will use Alexnet in the slides

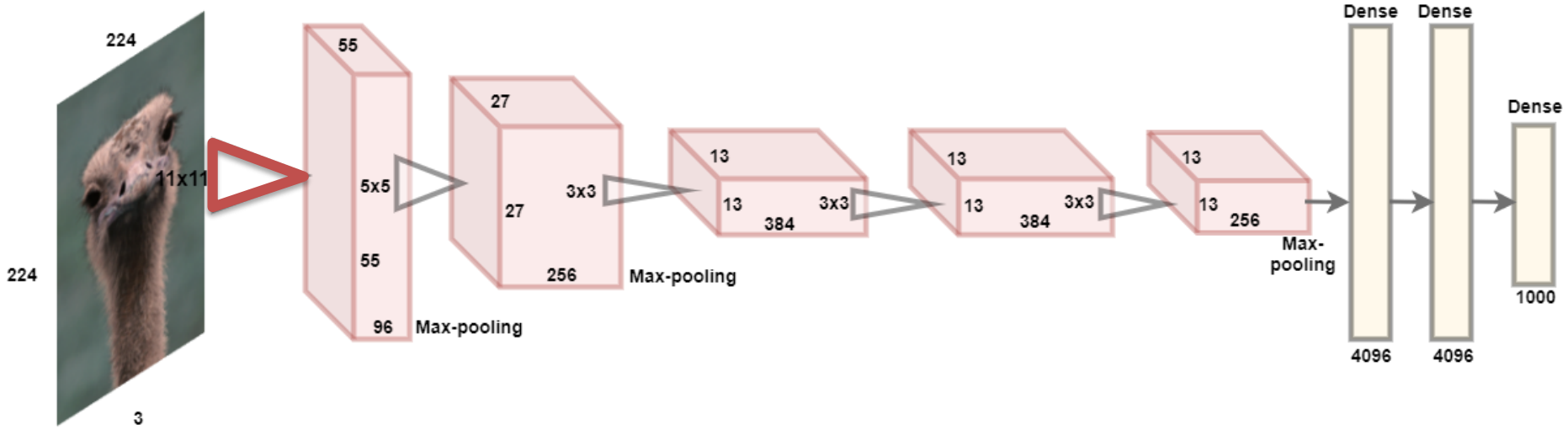
Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.

Deep Neural Networks Learn Hierarchical Feature Representations

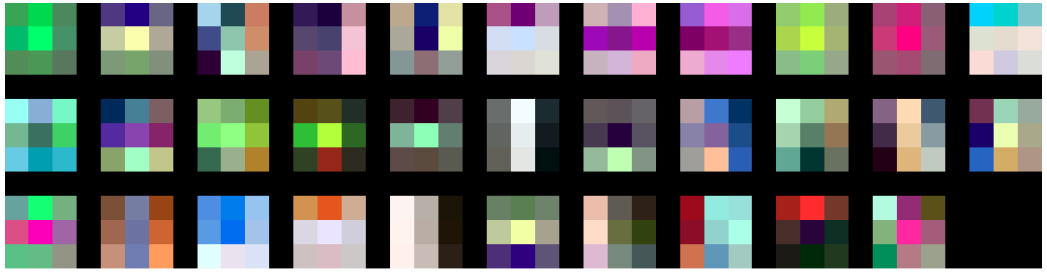




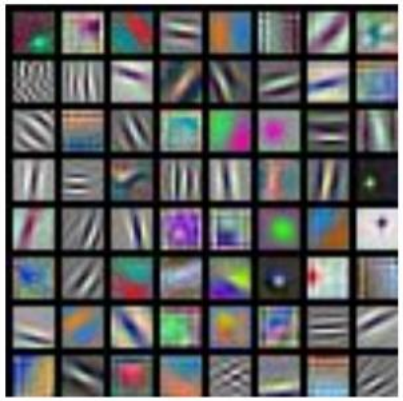
Visualizing Filters of First Layer



Visualizing Filters of First Layer



Mobilenet-v2 (3x3x3x32)

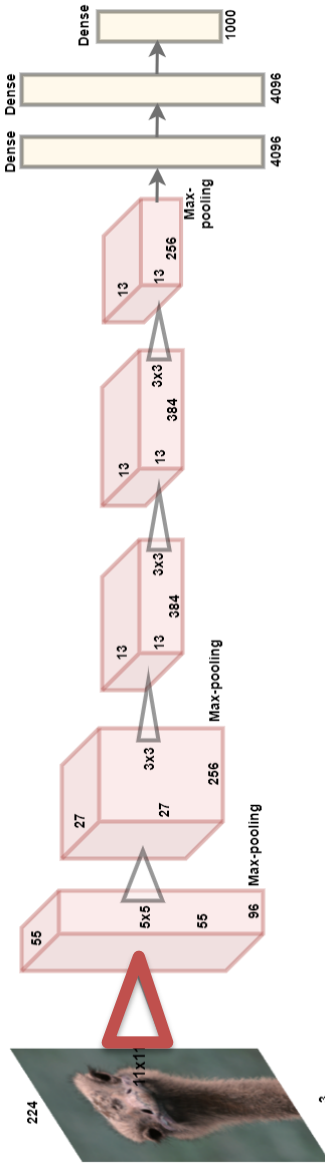


AlexNet:
64 x 3 x 11 x 11

ResNet-18:
64 x 3 x 7 x 7

ResNet-101:
64 x 3 x 7 x 7

DenseNet-121:
64 x 3 x 7 x 7



Visualizing Filters of Intermediate Layers

- Intermediate filters are hard to interpret for the humans
- We will come back to it in the end to see effective techniques for intermediate filters visualization

Weights:

Layer1: 16x3x7x7

Weights:

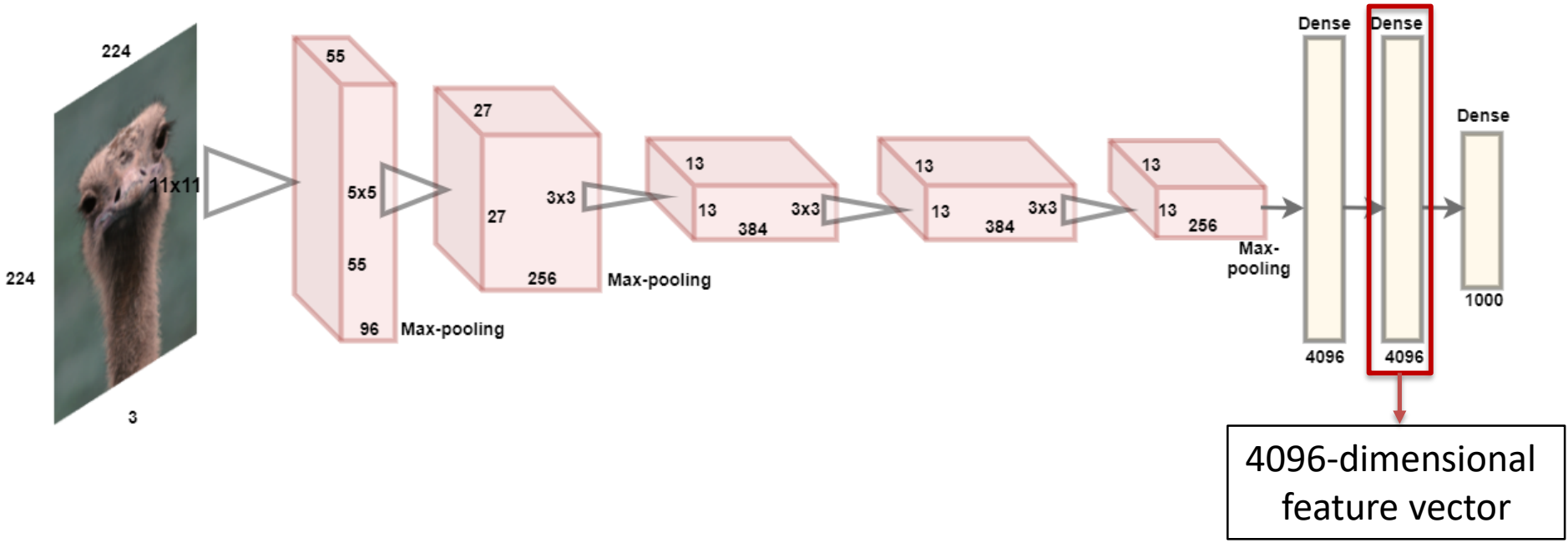
Layer2: 20x16x7x7

Weights:

Layer3: 20x20x7x7

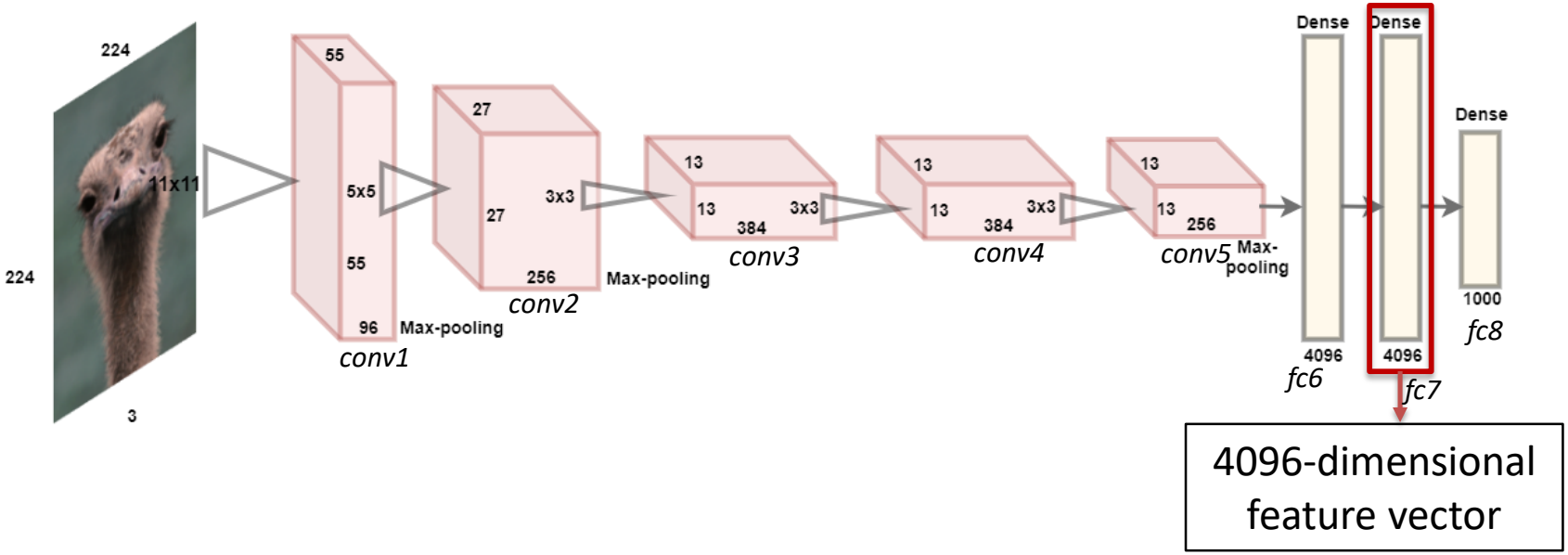
Weights visualization from *: https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html*

Interpreting the Last Layer



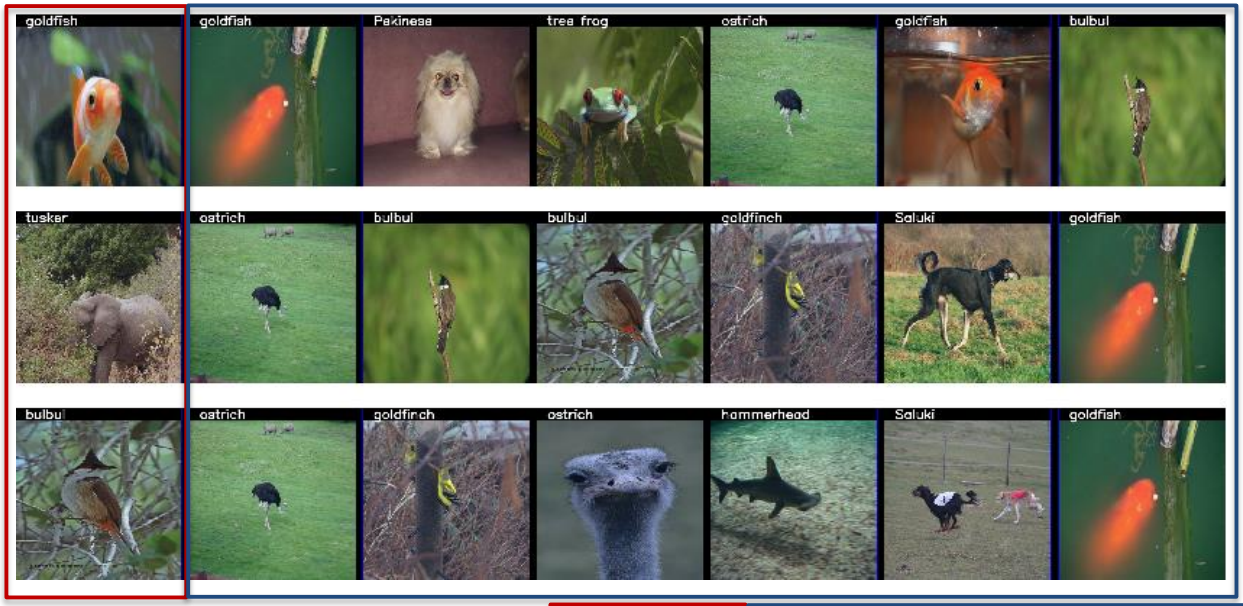
Interpreting the Last Layer

Using K Nearest Neighbors (KNN) to visualize the images in feature space



- Collect *fc7* feature vector (4096-dimensional) for all the images from the dataset ; create a KNN classifier using these feature vectors
- Take a test image, compute its *fc7* feature and get top k nearest neighbors from the KNN classifier
- Observe the difference between:
  - KNN classifier constructed using images
  - KNN classifier constructed using features

K-Nearest Neighbors

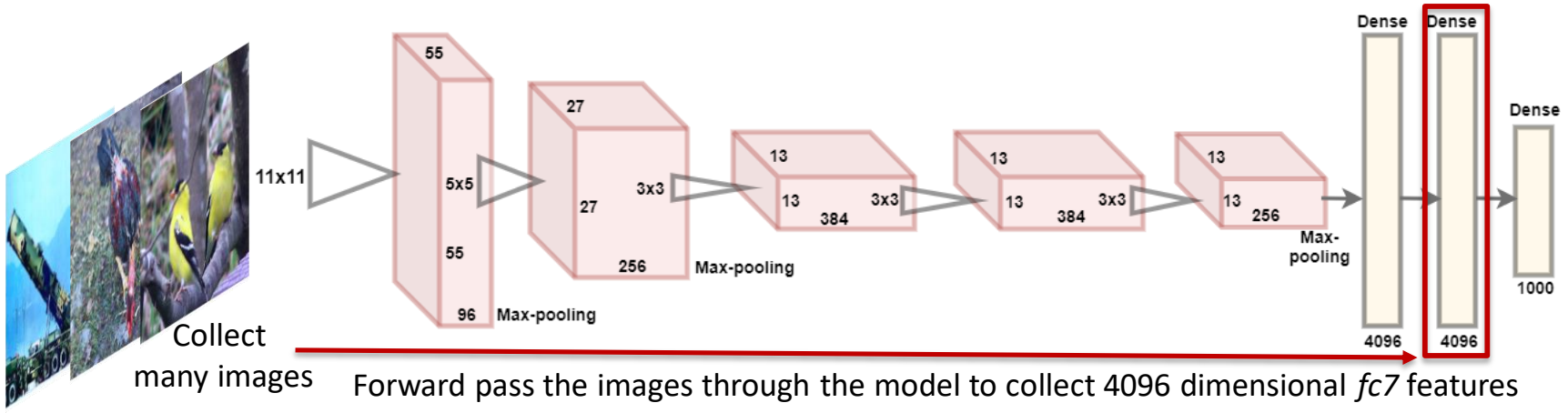


Pixel space
(384x384x3 Dimensional)

Feature Space
(4096 dimensional)

Interpreting the Last Layer

Using Principal Component Analysis (PCA) to project features on 2D space



Collect many images

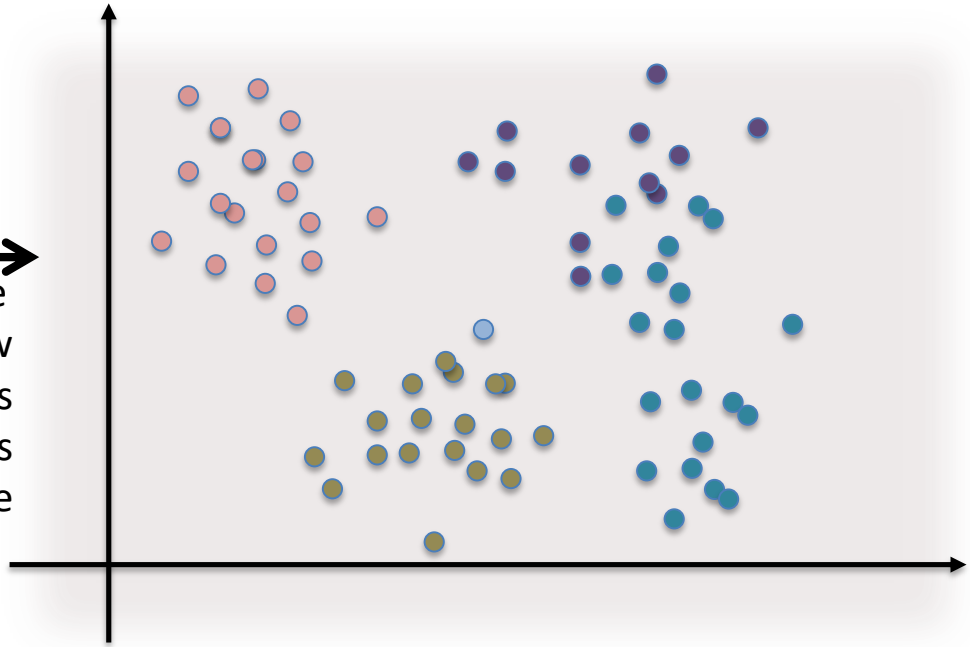
Forward pass the images through the model to collect 4096 dimensional *fc7* features

0.32	0.32	0.32	…	0.32
0.21	0.21	0.21		0.21
0.18	0.18	0.18		0.18
0.34	0.34	0.34		0.34
⋮	⋮	⋮		⋮
0.01	0.01	0.01	…	0.01
0.08	0.08	0.08		0.08

nx4096

PCA

Project to 2D Space to visualize how well the features for each class distributed on the 2D space



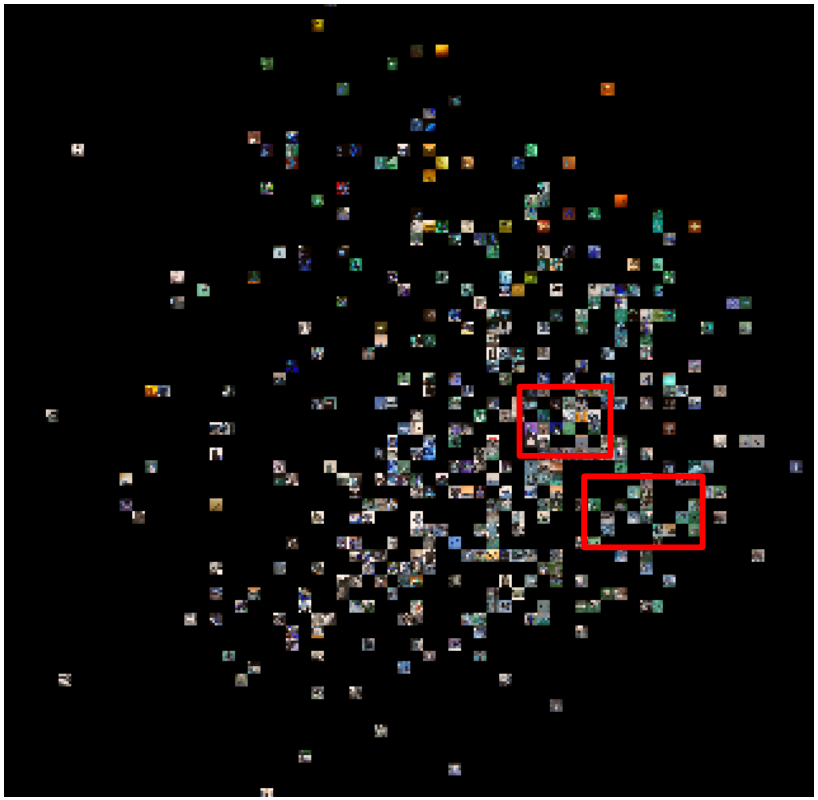
Interpreting the Last Layer

Using Principal Component Analysis (PCA) to project features on 2D

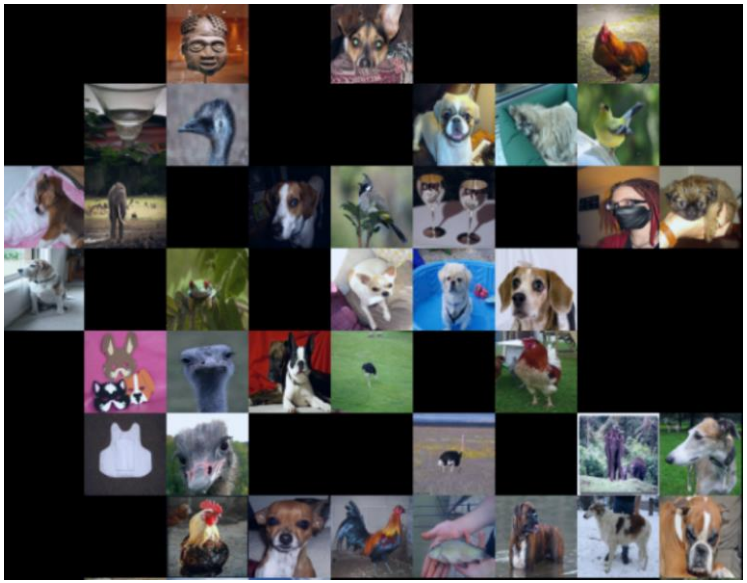
Images from: *https://cs.stanford.edu/people/karpathy/cnnembed/*

Interpreting the Last Layer

Using Principal Component Analysis (PCA) to project features on 2D

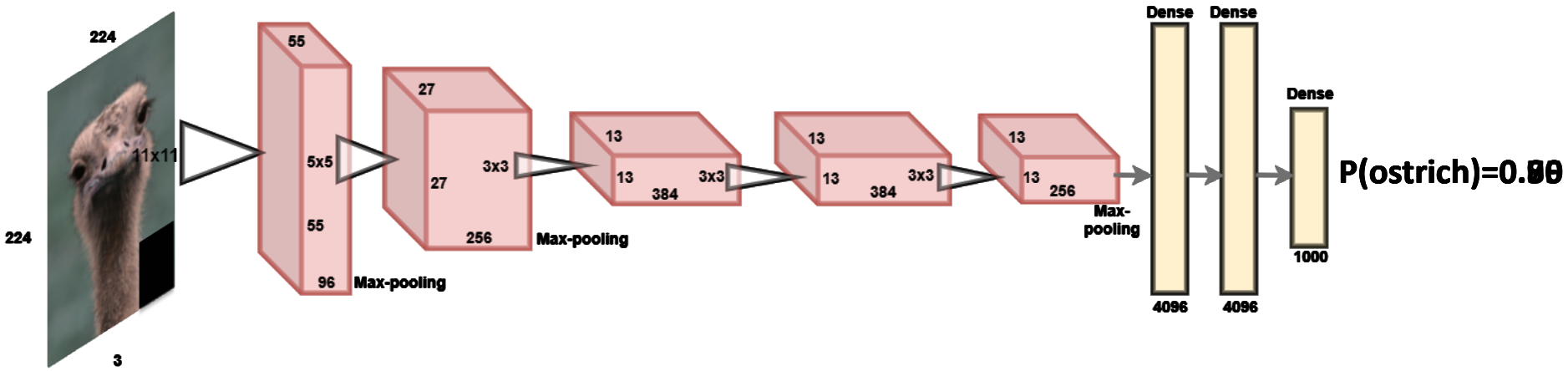


Our Experimental Dataset

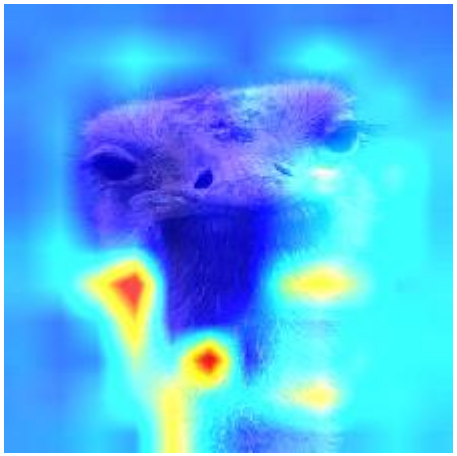


Which Image Pixels Affect the Output ?

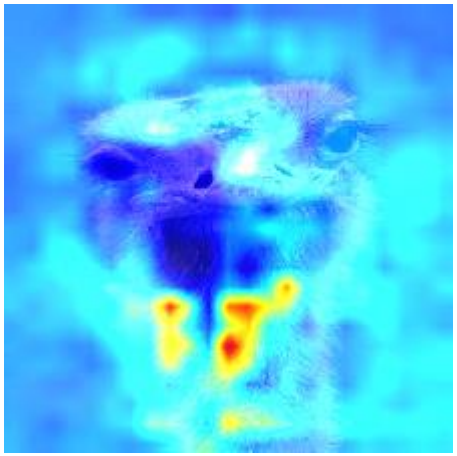
Saliency Maps via Occlusion



Saliency Map



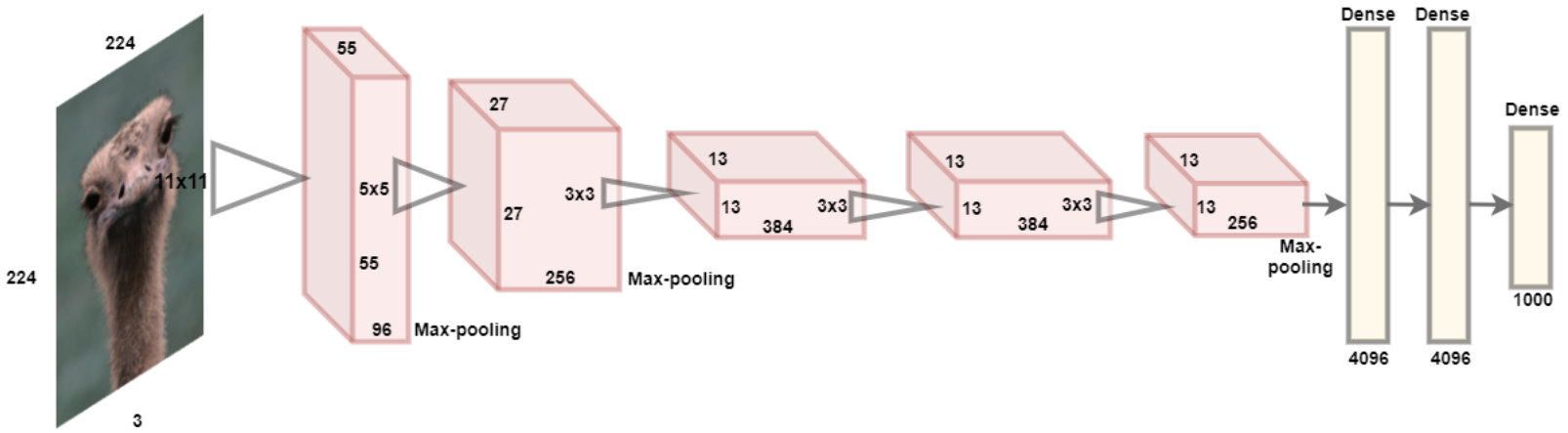
Patch size 20x20



Patch size 10x10

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

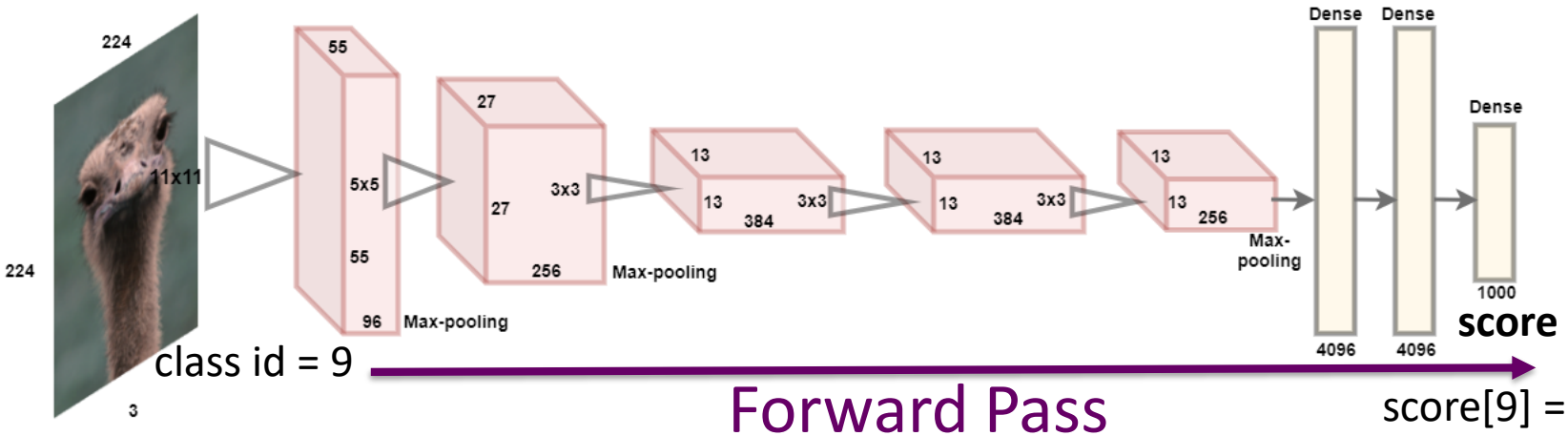
Saliency Maps via Occlusion



Drawback: Computationally Demanding

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Saliency Maps via Gradients



- Compute gradient of class score w.r.t image pixels
- Take absolute value
- Compute max along the channel

$$im\_grad = \frac{\partial score[9]}{\partial im_{x,y,c}}$$

Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop, 2014

Saliency Maps via Gradients

As seen before, the syntax to compute the gradients of model parameters w.r.t loss

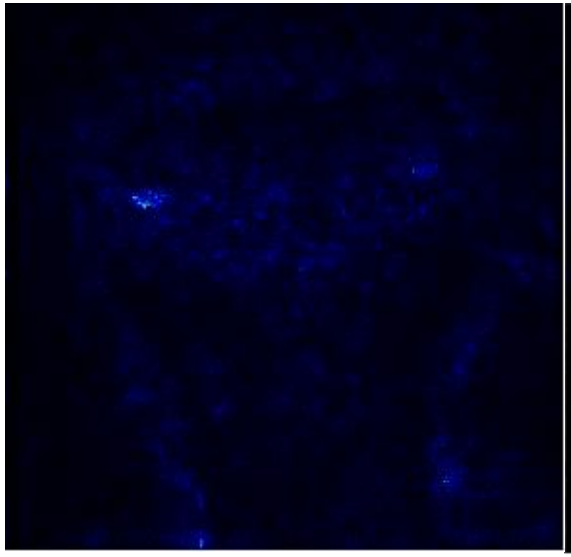
```
with tf.GradientTape() as tape:
    outp = model(inp, training=True)
    loss = tf.reduce_mean(tf.losses.sparse_categorical_crossentropy(labels, outp))
    gradients = tape.gradient(loss, model.trainable_variables)
```

Similarly, we can compute the gradient of the image pixels w.r.t true scores

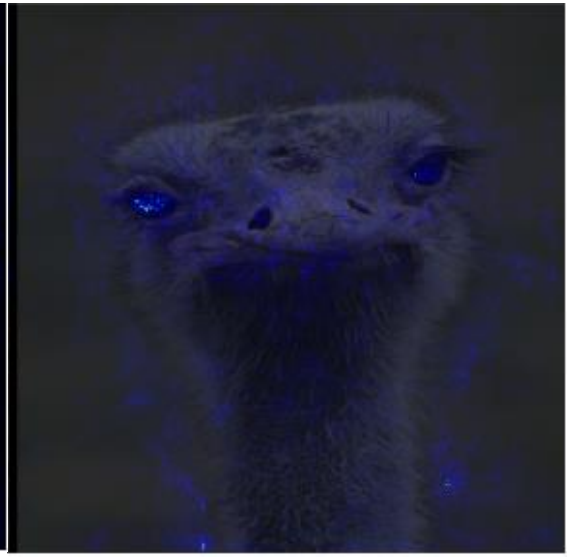
```
with tf.GradientTape() as tape:
    X = tf.Variable(X)
    # forward pass
    scores = model(X, training=False)
    # extract scores corresponding to true labels
    # compute the gradient w.r.t image pixels
    grad_X = tape.gradient(sc, X)
# take absolue value and max along channels -> saliency
return saliency.numpy()
```

Saliency Maps via Gradients

Input Image



Saliency Map

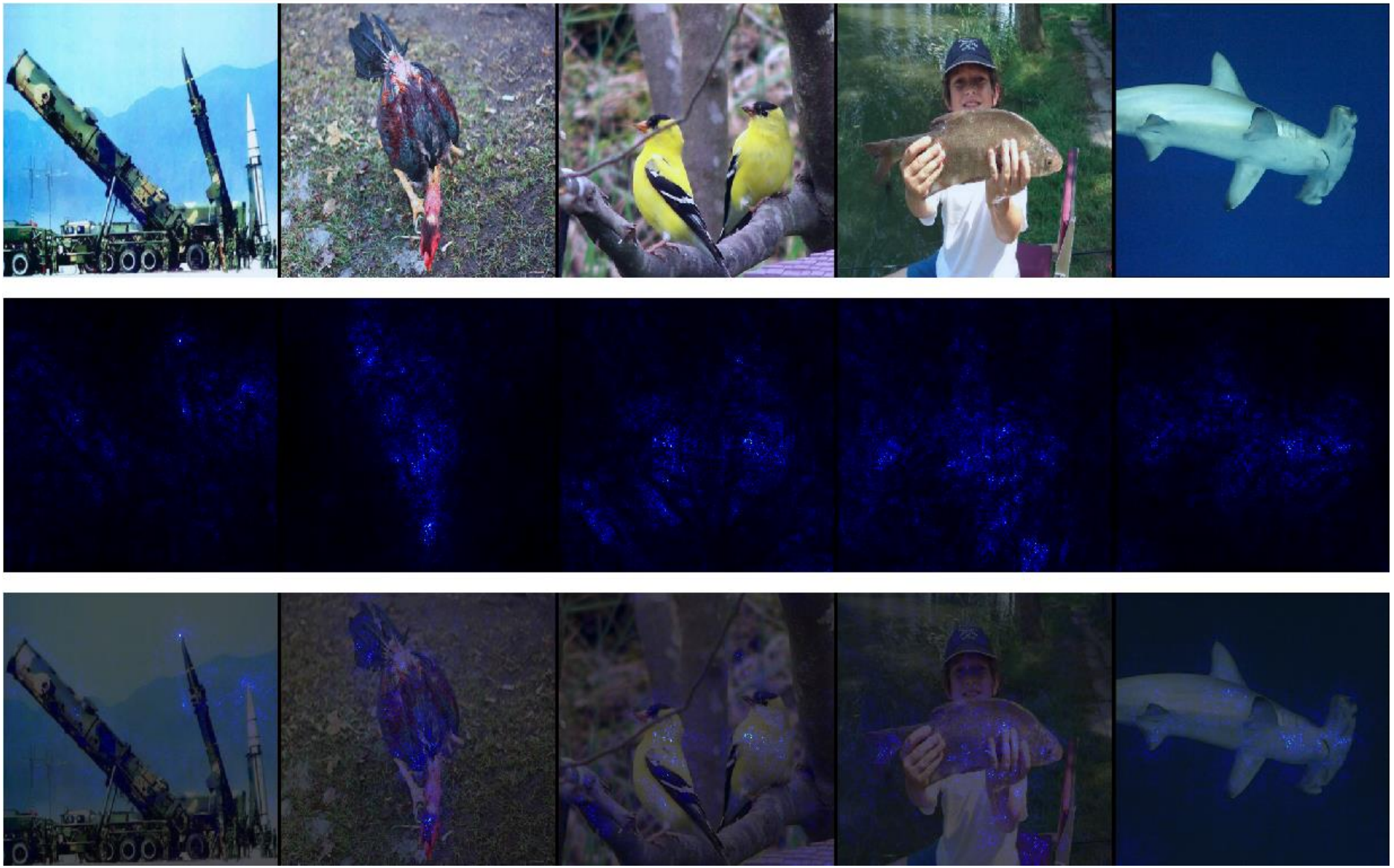


Saliency Map Overlaid on the Input Image

$$im\_grad = \frac{\partial score[9]}{\partial im_{x,y,c}}$$

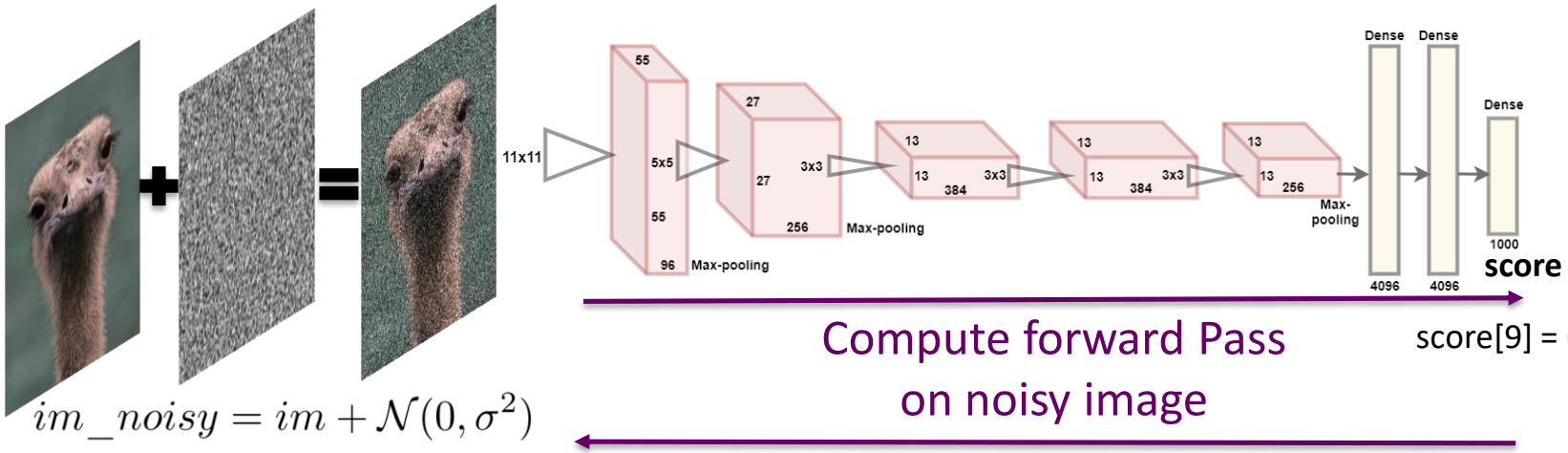
Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop, 2014

Saliency Maps via Gradients



Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop, 2014

Saliency Maps via Smooth Gradients



- Repeat many times, accumulate gradient and take the average

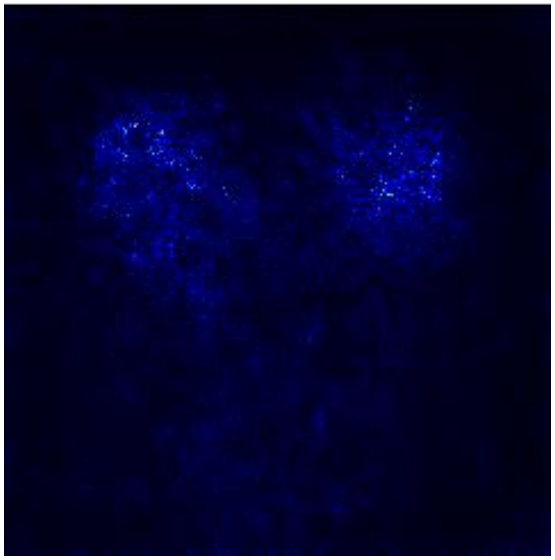
$$im\_grad = \frac{1}{n}\sum_{1}^{n}\frac{\partial score[9]}{\partial im\_noisy_{x,y,c}}$$

- Take absolute value
- Compute max along the channel

Smilkov D, Thorat N, Kim B, Viégas F, Wattenberg M. "Smoothgrad: removing noise by adding noise". arXiv:1706.03825. 2017

Saliency Maps via Smooth Gradients

Input Image



Smooth-grad Saliency Map

Smooth-grad Saliency Map with Overlaid on the Input Image

$$im\_grad = \frac{1}{n}\sum_{1}^{n}\frac{\partial score[9]}{\partial im\_noisy_{x,y,c}}$$

Smilkov D, Thorat N, Kim B, Viégas F, Wattenberg M. "Smoothgrad: removing noise by adding noise". arXiv:1706.03825. 2017

Saliency Maps via Smooth Gradients

Smilkov D, Thorat N, Kim B, Viégas F, Wattenberg M. "Smoothgrad: removing noise by adding noise". arXiv:1706.03825. 2017

Simple Gradients vs Smooth Gradients

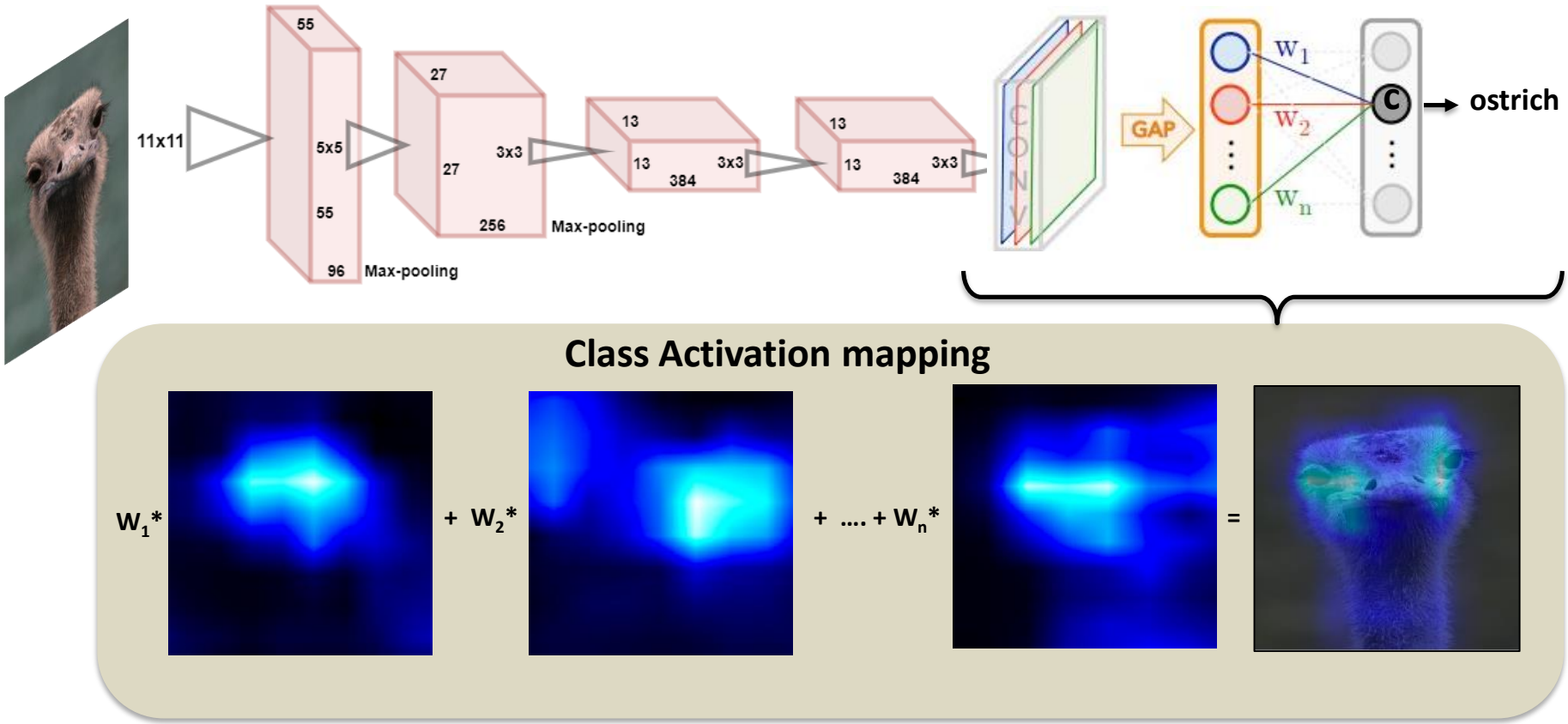
simpleGrad

smoothGrad

$$im\_grad = \frac{\partial score[9]}{\partial im_{x,y,c}}$$

$$im\_grad = \frac{1}{n}\sum_{1}^{n}\frac{\partial score[9]}{\partial im\_noisy_{x,y,c}}$$

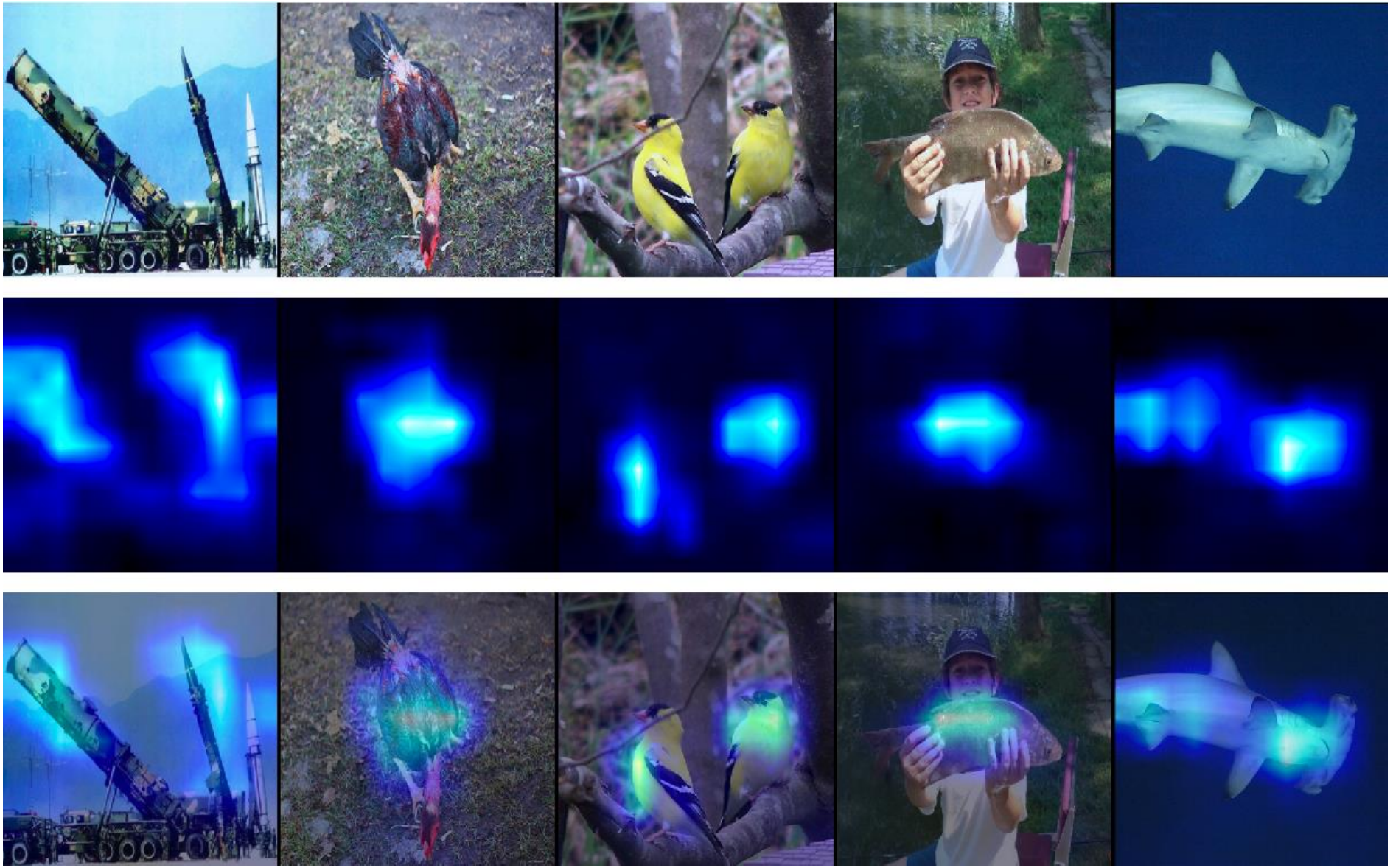
Class Activation Maps



$$S_c = \sum_k w_k^c \sum_{x,y} f_k(x,y) = \sum_{x,y} \sum_k w_k^c f_k(x,y)$$

Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR, 2016

Class Activation Maps



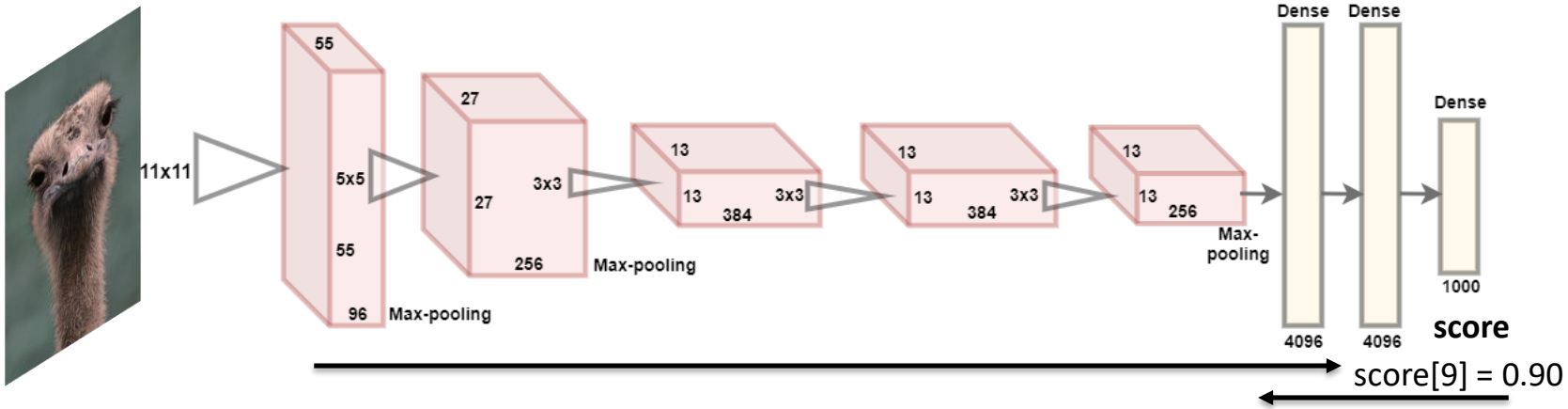
Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR, 2016

Class Activation Maps

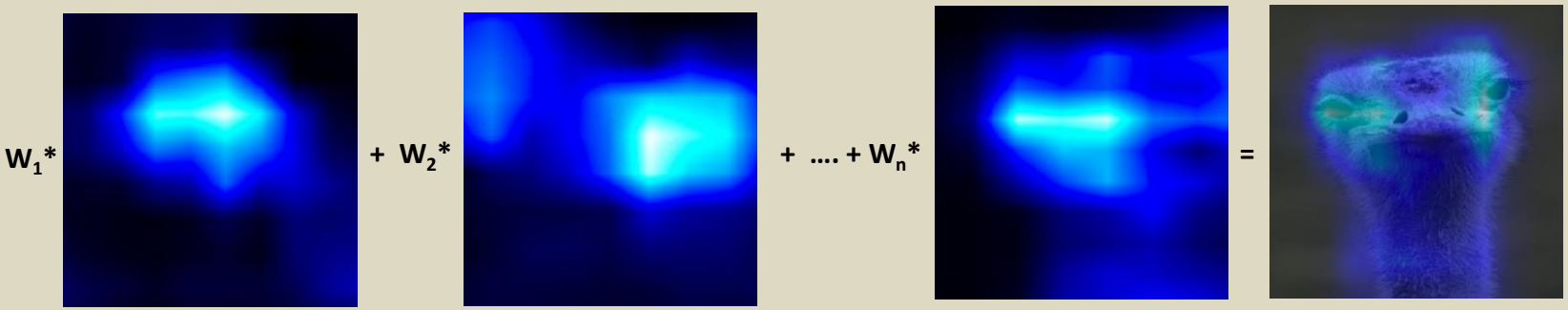
Drawback: Need a Global Averaging Pooling Layer

Grad-CAM

Incorporating Gradients in Class activation maps



Class Activation mapping

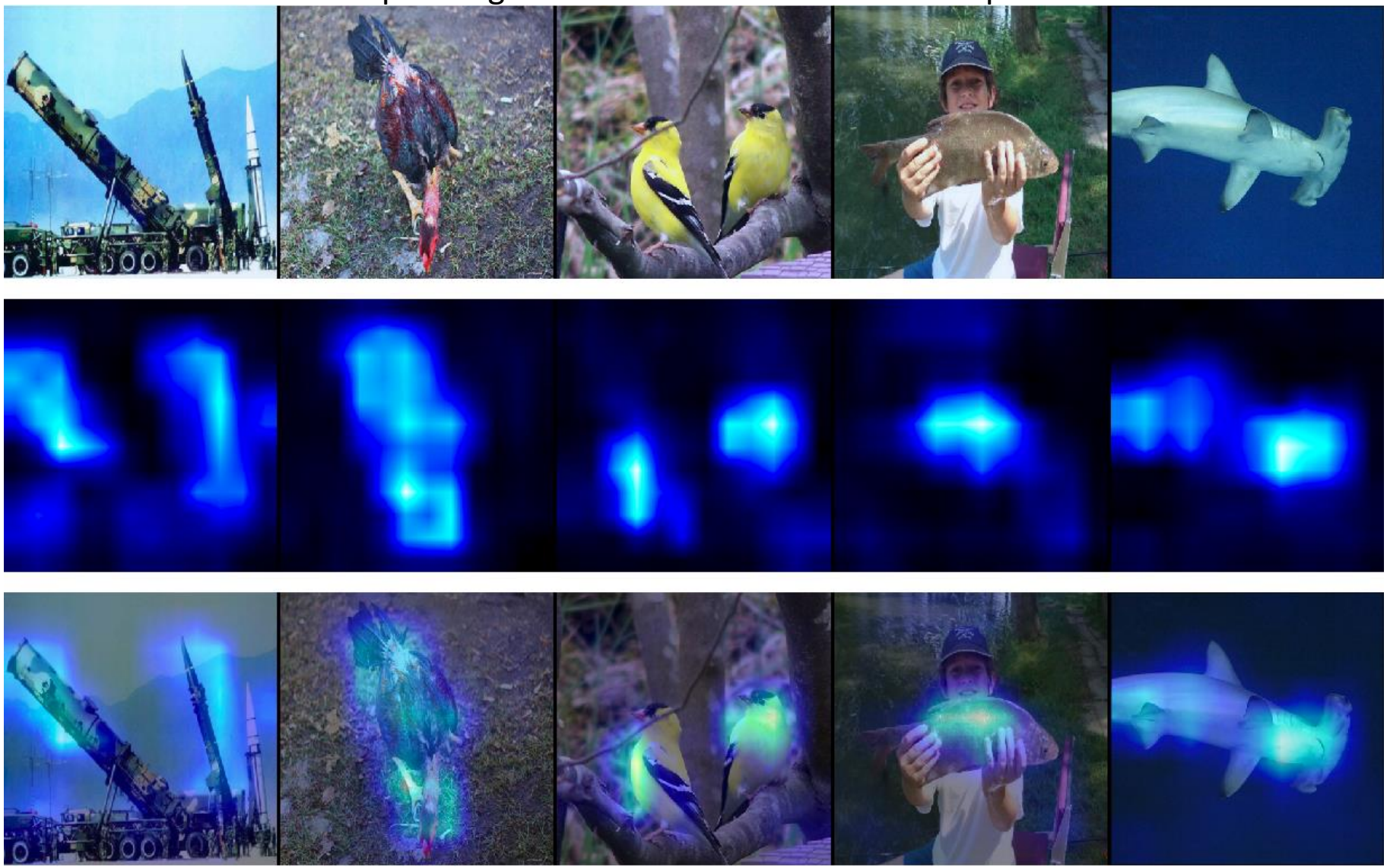


Compute W_i by averaging the gradients (No GAP)

Selvaraju, Ramprasaath R., et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization." ICCV, 2017

Grad-CAM

Incorporating Gradients in Class activation maps



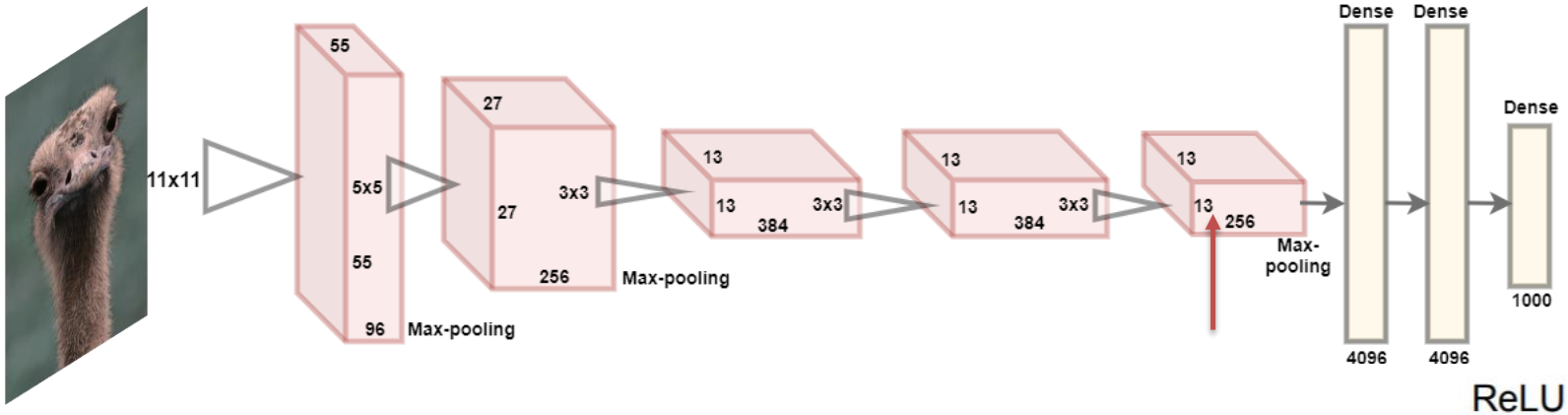
Selvaraju, Ramprasaath R., et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization." ICCV, 2017

CAM vs Grad-CAM

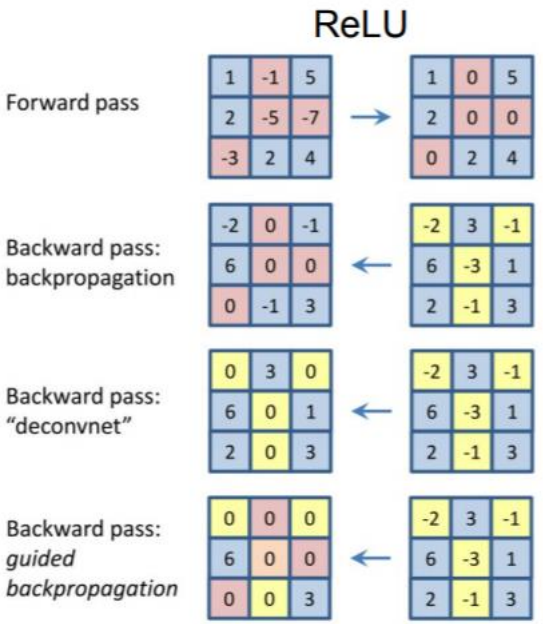
CAM

GradCAM

Visualizing Intermediate Features



- Pick a single intermediate neuron, e.g. one value in 256x13x13 *conv5* feature map
- Compute gradient of neuron value w.r.t image pixels

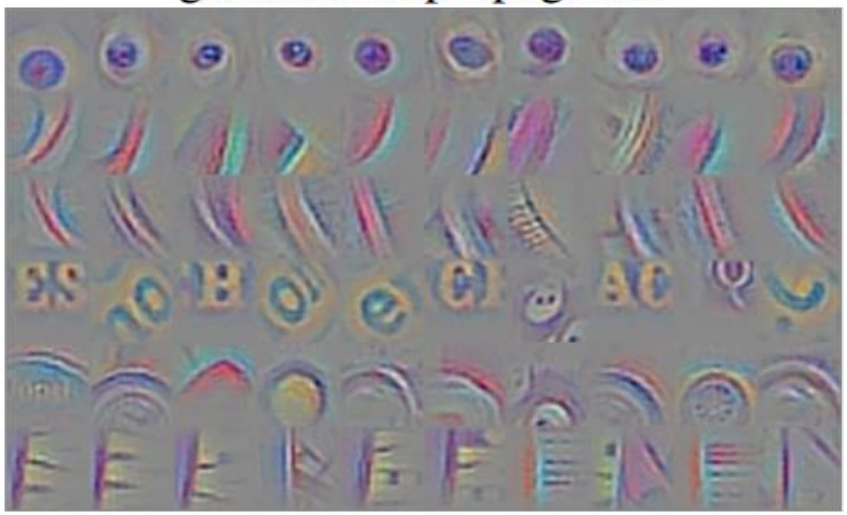


Use guided RELU for better visualization (Guided Backpropagation)

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015

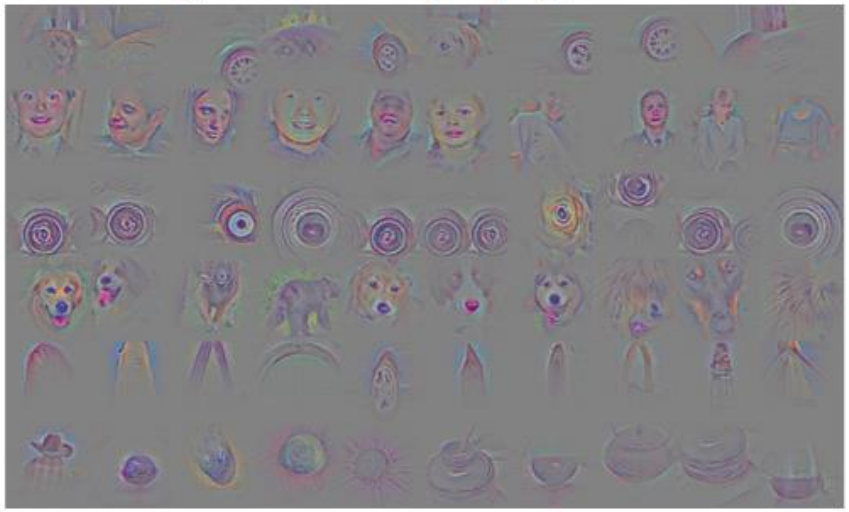
Visualizing Intermediate Features

guided backpropagation

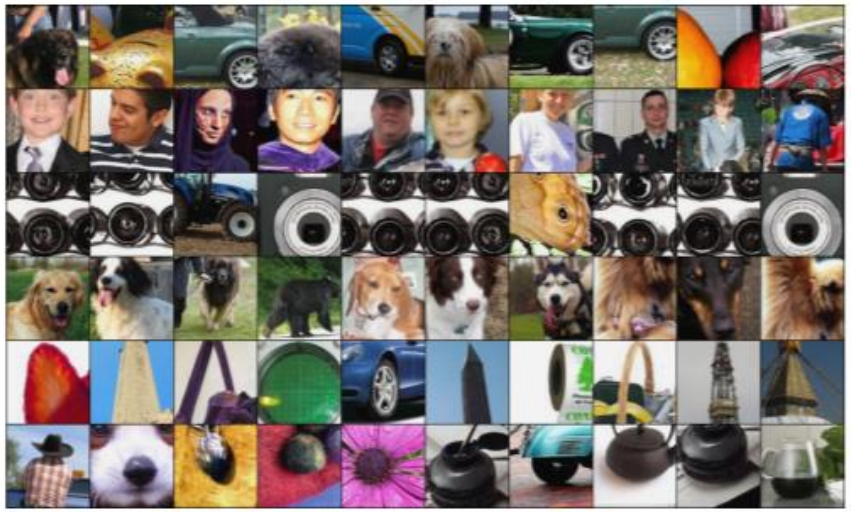


corresponding image crops

guided backpropagation

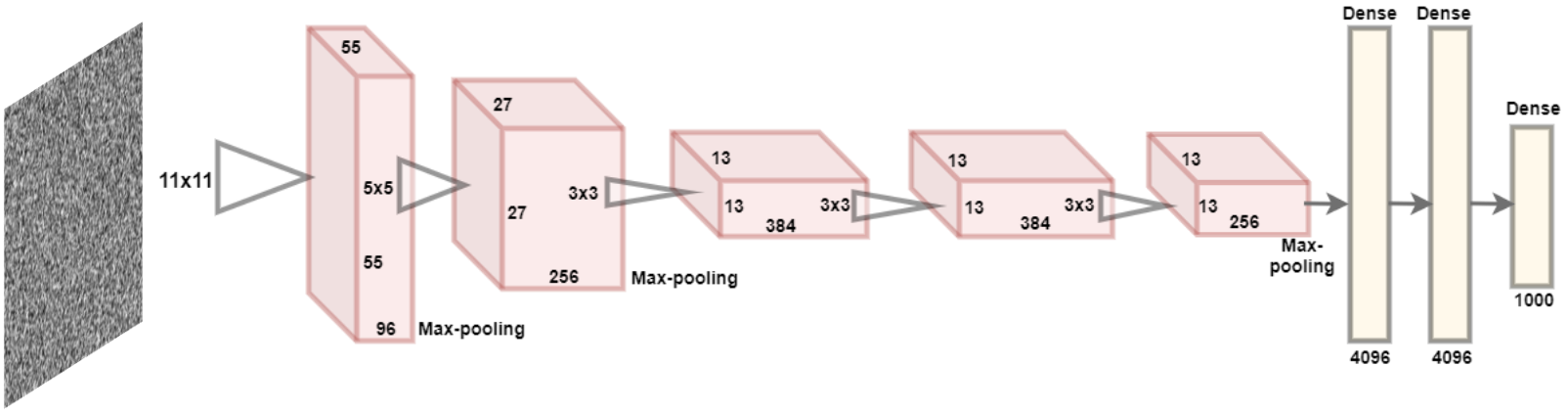


corresponding image crops



Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015

Visualizing per-class Features

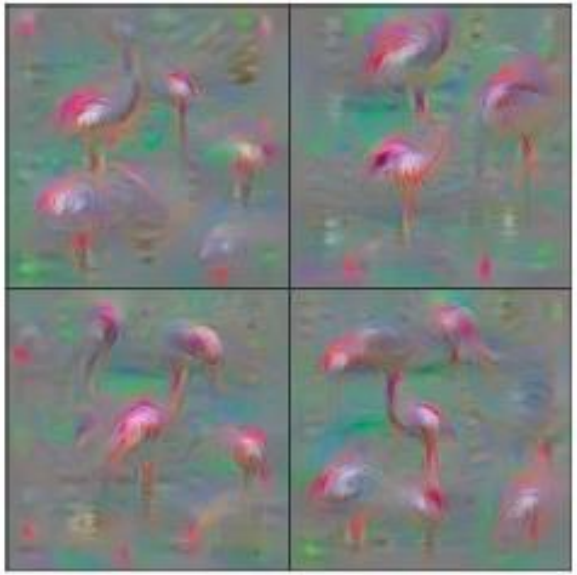


$$\arg\max_I S_c(I) - \lambda \|I\|_2^2$$

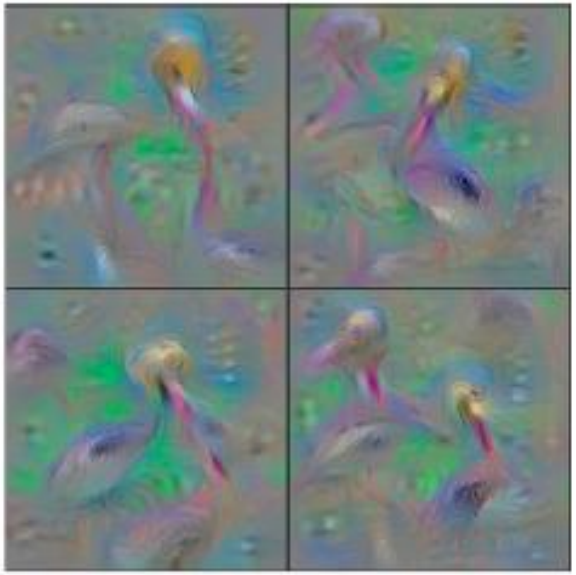
score for class c (before Softmax)

- Initialize random image
- Repeat:
  - Forward image to compute current scores
  - Backprop with L2 regularizer to get gradient of neuron value w.r.t image pixels
  - Make a small update to the image
  - Periodically, gaussian blur image, clip pixels with small values to zero, clip pixels with small gradients to zero

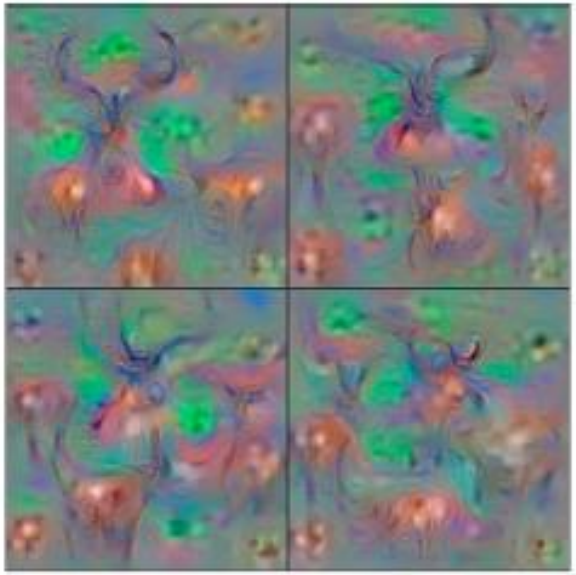
Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.



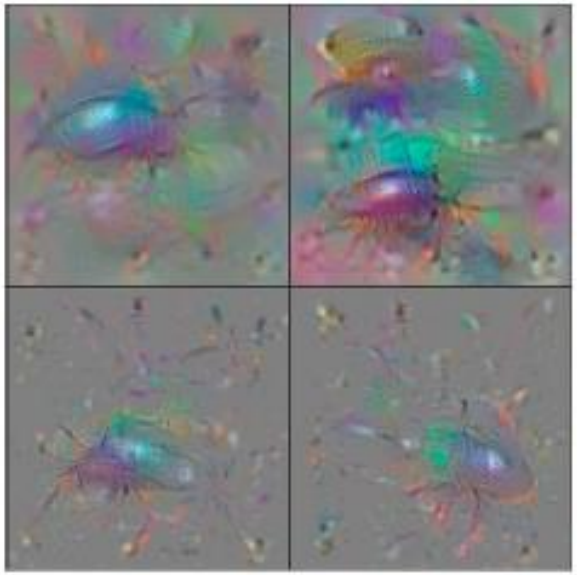
Flamingo



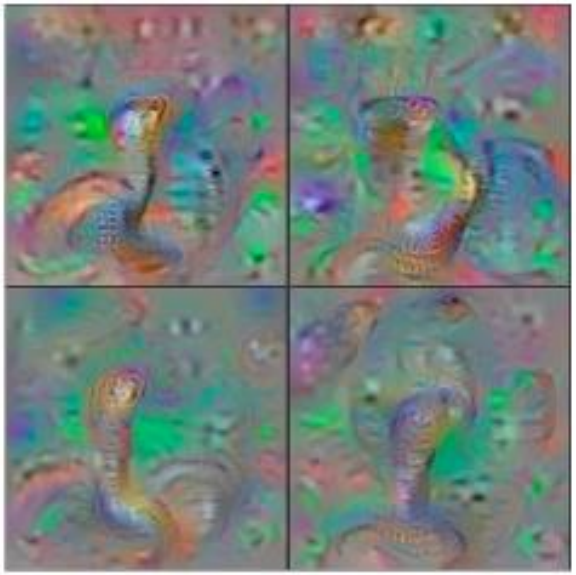
Pelican



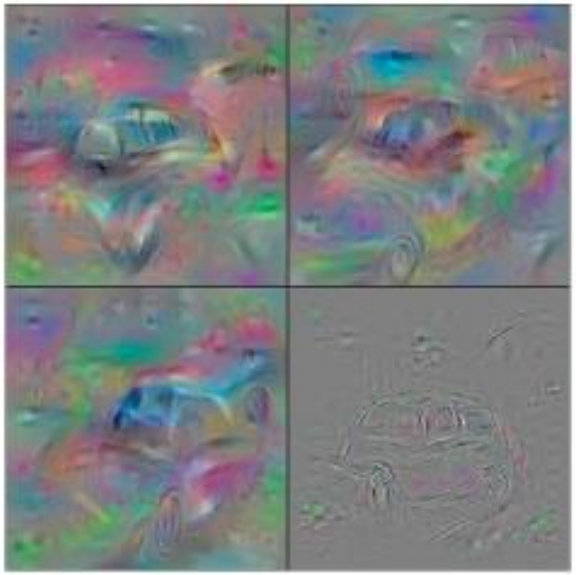
Hartebeest



Ground Beetle



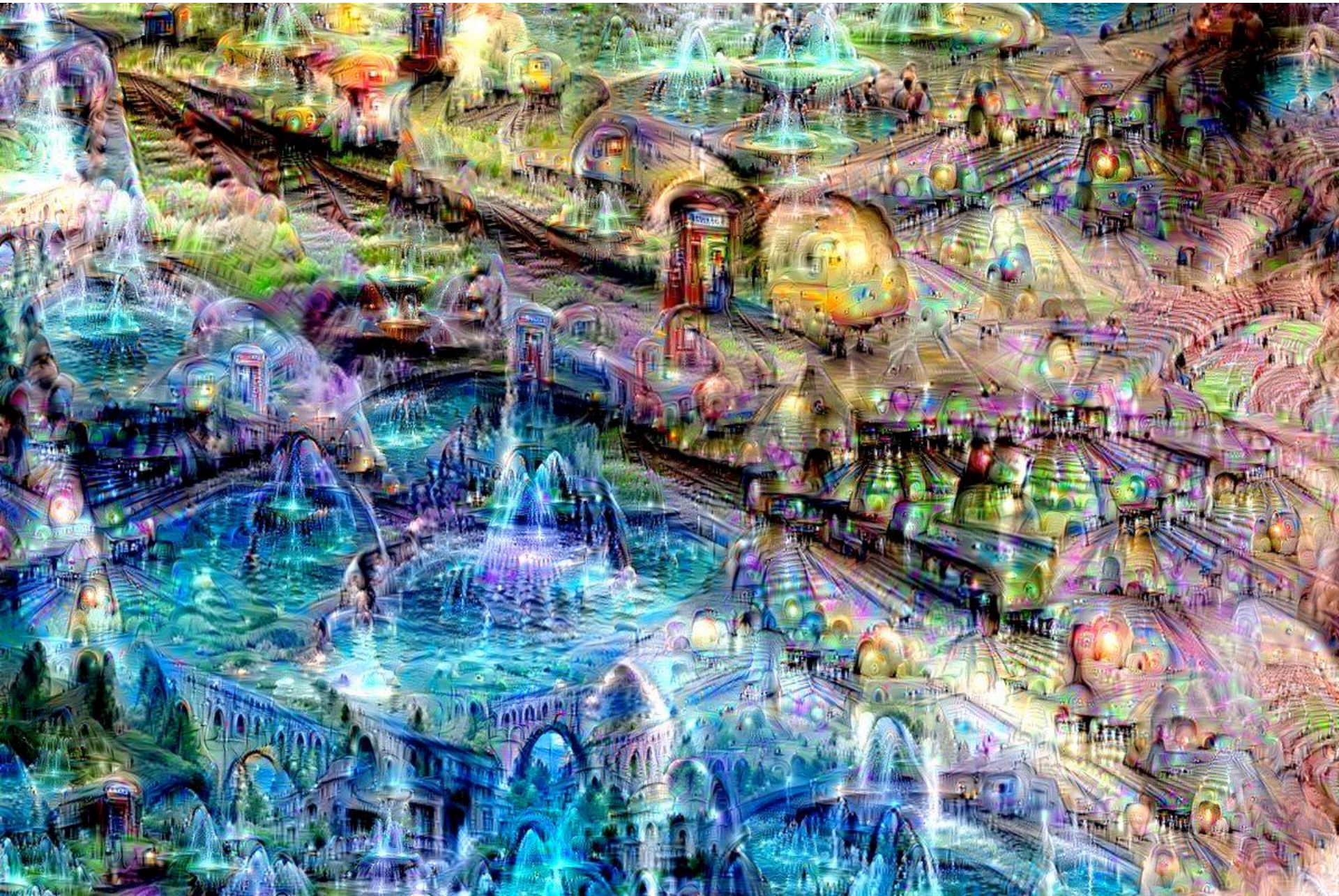
Indian Cobra



Station Wagon

Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

Deep Dream



The End